Efficient Pruning of Probabilistic Automata[1]

Franck Thollard and Baptiste Jeudy

Université de Lyon,
Université Jean-Monnet,
Laboratoire Hubert Curien UMR CNR 5516,
{franck.thollard,baptiste.jeudy}@univ-st-etienne.fr

Abstract. Applications of probabilistic grammatical inference are limited due to time and space consuming constraints. In statistical language modeling, for example, large corpora are now available and lead to managing automata with millions of states. We propose in this article a method for pruning automata (when restricted to tree based structures) which is not only efficient (sub-quadratic) but that allows to dramatically reduce the size of the automaton with a small impact on the underlying distribution. Results are evaluated on a language modeling task.

1 Introduction

Probabilistic automata have proved to be very useful in many fields. Among these, we can note natural language processing, e.g. machine translation [1], character recognition [2], ... Unfortunately, some particular tasks – such as language modeling for speech recognition – cannot be achieved because of the time complexity of the algorithms. For example, alergia [3], acyclic-infer [2], MDI [4], DDSM [5], multinomial-infer [6] have a worst case quadratic complexity.

In these grammatical inference algorithms, an automaton representing the data is built as a first step. It is then generalized using state merging operations. In this second step, pairs of states are considered and merged if they are sufficiently close to be considered equivalent. The algorithms differ in the strategy they use to choose the pair of states to consider and by the equivalence criteria.

We propose in this article to prune the probabilistic automaton obtained in the first step by deleting some – carefully selected – states. Our goal is to get a much smaller automaton on which the inference will be efficient.

Of course, the pruned automaton is only an approximation of the data. The deleted states are therefore chosen to minimize the distance between the distributions before and after the pruning.

The next section provides the notations. Section 3 describes in more details the problem we address in Sect. 4. A new algorithm is presented in Sect. 5 which is evaluated in Sect. 6. We then conclude.

2 Definitions and Notations

Let Σ be a *finite alphabet* and $\Sigma^\star$ (resp. Σ^+) be the set of all strings that can be built from Σ, including (resp. not including) the empty string denoted by λ. We denote $\Sigma' = \Sigma \cup \{\lambda\}$.

[1] This work was supported by the BINGO2 project (ANR-07-MDCO 014-02).

A *language* is a subset of $\Sigma^\star$. By convention, symbols in Σ will be denoted by letters from the beginning of the alphabet $(a, b, c, ...)$ and strings in $\Sigma^\star$ will be denoted by end of the alphabet letters $(..., x, y, z)$.

A *stochastic language* $\mathcal{D}$ is a probability distribution over $\Sigma^\star$. We denote by $P_\mathcal{D}(x)$ the probability of a string $x \in \Sigma^\star$ under the distribution $\mathcal{D}$ or $P_{\mathcal{D}_\mathcal{A}}(x)$ if the distribution is modeled by a syntactic machine $\mathcal{A}$. The distribution must verify $\sum_{x\in\Sigma^\star} P_\mathcal{D}(x) = 1$.

A *sample* S is a multi-set of strings: as samples are usually built through sampling, one string may appear more than once. The number of times a string x appears in S is its *multiplicity* and is denoted $|S|_x$. The *cardinality* of sample S is the total number of strings in the sample (each counted with its multiplicity): $|S| = \sum_{x\in S} |S|_x$. The empirical finite-support distribution associated with S will be denoted as $\mathcal{D}_S$ and is defined by: $P_{\mathcal{D}_S}(x) = |S|_x/|S|$.

2.1 Probabilistic Automata

Definition 1. *A Deterministic Probabilistic Finite state Automaton* DPFA *is a tuple* $\mathcal{A}=\langle Q_\mathcal{A}, \Sigma, \delta_\mathcal{A}, q_0, p_\mathcal{A}\rangle$*, where* $Q_\mathcal{A}$ *is a finite set of* states*;* $q_0 \in Q_\mathcal{A}$ *is the* initial state*;* Σ *is* the alphabet*;* $\delta_\mathcal{A} : Q_\mathcal{A} \times \Sigma \to Q_\mathcal{A}$ *is a* transition function*;* $p_\mathcal{A} : Q_\mathcal{A} \times \Sigma' \to \mathbb{R}^+$ *are* transition probabilities *such that*

$$\forall q \in Q_\mathcal{A}, \qquad \sum_{a\in\Sigma'} p_\mathcal{A}(q, a) \;=\; 1. \tag{1}$$

For each state q, the probability $p_\mathcal{A}(q, \lambda)$ is not associated with a transition. It represents the probability that the string ends at q. Functions $\delta_\mathcal{A}$ and $p_\mathcal{A}$ are extended recursively from Σ to $\Sigma^\star$.

The probability of a string s according to a DPFA $\mathcal{A}$ is then defined as: $P_\mathcal{A}(s) = p_\mathcal{A}(q_0, s) \times p_\mathcal{A}(\delta_\mathcal{A}(q_0, s), \lambda)$ if $s \neq \lambda$ and $P_\mathcal{A}(\lambda) = p_\mathcal{A}(q_0, \lambda)$ else. If (1) holds, these probabilities define a probability distribution $\mathcal{D}_\mathcal{A}$ over $\Sigma^\star$.

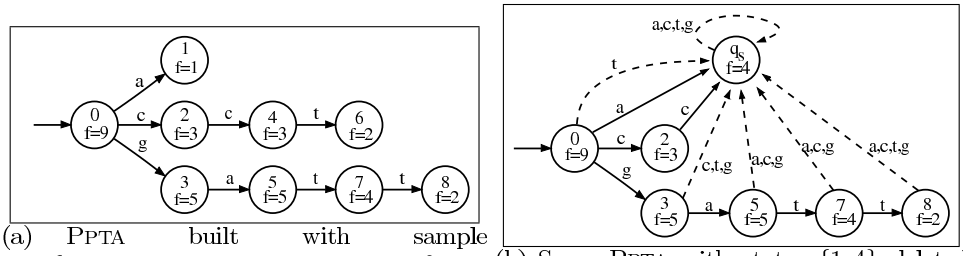
For a state q, $\Gamma^+(q)$ is the set $\{a \in \Sigma : p(q, a) \neq 0\}$ of the letters labeling the outgoing transitions. We also define the set of the descendants of q as $\text{desc}(q) = \{p :$ there is a path from q to p in $\mathcal{A}\} \bigcup \{q\}$ (we consider that $q \in \text{desc}(q)$). This definition can be extended to a set of states D: $\text{desc}(D) = \bigcup_{q\in D} \text{desc}(q)$. Finaly, the size of a DPFA $\mathcal{A}$, noted $|\mathcal{A}|$, is defined as its number of states.

2.2 Probabilistic Prefix Trees

Definition 2. *A* probabilistic prefix tree automaton PPTA *is a particular case of* DPFA *where the underlying graph (defined by the states and the non zero probability transitions) is a tree rooted at the initial state* q_0*.*

A PPTA built from a sample S (denoted PPTA_S) is the prefix tree built on S. Transitions which are not in the prefix tree are added as loops with zero probabilities. Other probabilities are estimated using state frequencies: the state frequency of $q \in Q$ with respect to a sample S is defined as $\text{freq}_S(q) = \sum_{x\in\{uv\in S:\delta(q_0,u)=q\}} |S|_x$. The transition probability $p(q, a)$ is then estimated by $\text{freq}_S(\delta(q, a))/\text{freq}_S(q)$ and $p(q, \lambda)$ is computed in order to satisfy (1). Given this construction, the distribution induced by PPTA_S and $\mathcal{D}_S$ are identical. More details on probabilistic automata can be found in [7].

Figure 1(a) presents a PPTA built from the sample $S = \{a, cct, cct, cc, gatt, gat, gat, ga, gatt\}$.



(a) PPTA built with sample $S = \{a, cct, cct, cc, gatt, gat, gat, ga, gatt\}$

(b) Same PPTA with states $\{1, 4\}$ deleted and a smoothing state q_s added.

Fig. 1. A PPTA and a pruned smoothed PPTA. States are labeled by their numbers and their frequencies freq_S. Dashed arrows represent smoothed transitions.

2.3 Smoothing Probabilistic Automata

Smoothing discrete distributions is a widely studied field (see [8] for a survey). It has been addressed mainly in real world applications (speech recognition [8], information retrieval [9], ...) as it has shown to dramatically improve the estimate of discrete distributions. Moreover, smoothing must be taken into account from the theoretical point of view (see [2, 10] for some links between smoothing and learnability of probabilistic automata). We will therefore consider here smoothed automata.

When the main model cannot parse the input string, a back-off transition is dynamically built that goes to a back-off model (an unigram model) in which the end of the parsing is done. The probability $P_{\mathcal{U}}(a), a \in \Sigma \cup \{\lambda\}$ is estimated by $|a|/||S||$ where $|a|$ is the frequency of the letter a in S and $||S||$ the total number of symbols in S. This smoothing technique uses a parameter $0 < \varepsilon < 1$. The smoothed transition probability p_{smooth} is estimated by:

$$p_{\text{smooth}}(q, a) = \begin{cases} \frac{\text{freq}(\delta(q,a)) - \varepsilon}{\text{freq}(q)} & \text{if } a \in \Gamma^+(q) \\ K.P_{\mathcal{U}}(a) & \text{else} \end{cases} \tag{2}$$

$$p_{\text{smooth}}(q, \lambda) = \begin{cases} \frac{\left(\text{freq}(q) - \sum_{a \in \Gamma^+(q)} \text{freq}(\delta(q,a))\right) - \varepsilon}{\text{freq}(q)} & \text{if } p(q, \lambda) \neq 0 \\ K.P_{\mathcal{U}}(\lambda) & \text{if } p(q, \lambda) = 0 \end{cases} \tag{3}$$

where value K is chosen to ensure that (1) holds.

In practice, to apply this smoothing, the PPTAis modified in three steps:

1. a new smoothing state (called q_s) is added. For all a in Σ, $\delta(q_s, a) = q_s$ and for all a in Σ', $p(q_s, a) = P_{\mathcal{U}}(a)$;
2. all transitions with a zero probability in the PPTA are redirected to q_s;
3. the transition probability p is replaced by the smoothed one p_{smooth}.

If this smoothing state is added to a PPTA, the resulting automaton is no longer strictly speaking a PPTA. We call such an automaton a *smoothed* PPTA. In the following, all PPTAs are assumed to be smoothed.

Fig. 1(b) presents the smoothed automaton obtained after pruning states 1 and 4.

2.4 Deleting states

We will prune a given automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, p\rangle$ by removing a state q_d ($q_d \neq q_s$). In order to maintain a complete automaton, the incoming transitions of q_d are redirected to q_s. After the deletion of q_d, several states may become unreachable from the initial state and are also deleted. This set of unreachable states is denoted $U_{\mathcal{A}}(q_d)$. The formal definition of the automaton $\mathcal{A}_{\overline{q_d}} = \langle Q', \Sigma, \delta', q'_0, p'\rangle$ obtained from $\mathcal{A}$ by deleting q_d is then: $Q' = Q \setminus U_{\mathcal{A}}(q_d)$; $\delta'(q,a) = q_s$ if $\delta(q,a) = q_d$, else $\delta'(q,a) = \delta(q,a)$; $q'_0 = q_0$; p' is the restriction of p to Q'. Note that the obtained automaton still respects equation (1) and thus defines a stochatic language.

The deletion operation is commutative (*i.e.*, removing state q and then state q' is equivalent to removing state q' and then state q) and we can therefore define $\mathcal{A}_{\overline{D}}$, the automaton obtained from $\mathcal{A}$ by deleting a set of states D.

Given an automaton $\mathcal{A}$ and two sets of states D and D', if they have the same set of descendants (*i.e.*, $\mathrm{desc}(D) = \mathrm{desc}(D')$), then $\mathcal{A}_{\overline{D}} = \mathcal{A}_{\overline{D'}}$ (for instance, the PPTA of figure 1(b) can be obtained by deleting $D = \{1,4,6\}$ or $D = \{1,4\}$). In a PPTA, there is a minimal set D among those having the same descendants ($\{1,4\}$ in our example). We call this set a *cut set of state* and is formally defined as a set D such that for every q, q' in D, $q \notin \mathrm{desc}(q')$.

Next section presents the dissimilarity measure used to quantify the modification of the distribution induced by the pruning method.

2.5 Kullback-Leibler Divergence

The dissimilarity between two distributions will be evaluated through the Kullback-Leibler divergence [11]:

$$KL(\mathcal{D}, \mathcal{D}') = -\sum_{x \in \Sigma^\star} \mathcal{D}(x) \log \frac{\mathcal{D}(x)}{\mathcal{D}'(x)} \tag{4}$$

Although this divergence is not a metric, some nice properties hold, *e.g.* it is positive and it bounds the L_1 distance from above [12]. The divergence is not symmetric. One of the distribution thus takes the role of a reference one. In term of information theory, $KL(\mathcal{D}, \mathcal{D}')$ represents the number of bits one must pay by coding messages drawn according to $\mathcal{D}$ using an optimal code derived from $\mathcal{D}'$. In our case, the distribution represented by the PPTA will serve as the reference.

3 Problems Setting

Depending on the application, the PPTA can be very large (*e.g.,* several million states). When processing these models (*e.g.,* applying a grammatical inference algorithm, or using such model *as is*), the size of the model can be problematic (most of inference algorithms have a quadratic complexity). A reduction of the size of the model is thus needed.

We follow an approach consisting in doing lossy compression: we want to find a trade-off between the size of the automaton and divergence. We define the loss function between two automata $\mathcal{A}$ and $\mathcal{A}'$ by:

$$\mathcal{L}(\mathcal{A}, \mathcal{A}') = \frac{KL(\mathcal{A}, \mathcal{A}')}{|\,|\mathcal{A}| - |\mathcal{A}'|\,|}. \tag{5}$$

In our case, the automaton $\mathcal{A}'$ is obtained from $\mathcal{A}$ by deleting some states. We will focus on two optimization problems:

1. Find the cut set of states D such that $\mathcal{L}(\mathcal{A}, \mathcal{A}_{\overline{D}})$ is minimal.
2. In the previous problem, there is no control on the number of deleted states. In practice, we want to delete a significant number of states. The second problem is thus to find the optimal cut set of states D with the constraint that the total number of deleted states $|\text{desc}(D)|$ is above a given threshold.

4 Problems Solutions

In this section, the goal is to find efficiently the optimal set of states D. Given a sample S, the algorithm will proceed in two steps: first build the PPTA_S and then find the set D. In order to be efficient even with large automata, the numerator of (5) must be computed efficiently.

4.1 Computing the Divergence Between two Automata

Carrasco [13] proposed an efficient way to compute the divergence between two distributions when they are represented by two DPFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, p\rangle$ and $\mathcal{A}' = \langle Q', \Sigma, \delta', q'_0, p'\rangle$. The computation of $KL(\mathcal{A}, \mathcal{A}')$ is made by summing state divergences:

$$KL(\mathcal{A}, \mathcal{A}') = \sum_{q\in Q}\sum_{q'\in Q'} kl(q, q') \quad \text{with} \quad kl(q, q') = c_{q,q'} \sum_{a\in\Sigma'} p(q, a) \log \frac{p(q, a)}{p'(q', a)} \tag{6}$$

and

$$c_{q,q'} = \sum_{x\in L_{q,q'}} p(q_0, x) \text{ where } L_{q,q'} = \{x \in \Sigma^\star : \delta(q_0, x) = q \wedge \delta'(q'_0, x) = q'\} \tag{7}$$

If $\mathcal{A}$ and $\mathcal{A}_{\overline{\{q_1,q_4\}}}$ are the automata of the running example, we have $L_{7,7} = \{gat\}$, $L_{6,q_s} = \{cct\}$, $L_{5,2} = \emptyset$, $c_{7,7} = p_{\mathcal{A}}(q_0, gat) = \frac{5-\epsilon}{9} \times \frac{5-\epsilon}{5} \times \frac{4-\epsilon}{5}$ (where ϵ is the smoothing parameter, see (2), Sect. 2.3).

Equation (6) is very general and applicable to any automata. Its computation complexity is $O(|\mathcal{A}|.|\mathcal{A}'|.|\Sigma|)$ times the complexity of the computation of the coefficients $c_{q,q'}$. In his paper, Carrasco gives an iterative method that converges to the values of these coefficients.

4.2 Pruning Automata

In this section, (6) is simplified in the case where $\mathcal{A}' = \mathcal{A}_{\overline{D}}$ for some set of states D. The automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, p\rangle$ is a complete DPFA with a smoothing (or unigram) state denoted by q_s. We consider the deletion of a set of states D of $\mathcal{A}$ ($q_s \notin D$). The resulting DPFA is $A_{\overline{D}} = \langle Q', \Sigma, \delta', q'_0, p'\rangle$.

Lemma 1.

$$KL(\mathcal{A}, \mathcal{A}_{\overline{D}}) = \sum_{q\in desc(D)} kl(q, q_s) \tag{8}$$

The (omitted) proof of this lemma is based on the fact that most of the $kl(q, q')$ are zero because either $c(q, q') = 0$ or $p(q, a) = p'(q', a)$. The complexity of computing $KL(\mathcal{A}, \mathcal{A}_{\overline{D}})$ is therefore less than $O(|\mathcal{A}|.|\Sigma|)$ times the complexity of the computation of the coefficients c_{q,q_s}. However, we still need to find the optimal set of states D, and it is of course not possible to test all subsets of Q.

Another important point is that, in a PPTA, c_{q,q_s} does not depend on the set of deleted states D and therefore neither does $kl(q, q_s)$. It means that it is possible to compute $kl(q, q_s)$ for all q and then consider several possibilities for the cut set of states D without recomputing the kl values.

4.3 Solutions of the Optimization Problems

Given a cut set of states D and following (8), the loss function is now

$$\mathcal{L}(\mathcal{A}, \mathcal{A}_{\overline{D}}) = \frac{KL(\mathcal{A}, \mathcal{A}_{\overline{D}})}{|\mathcal{A}| - |\mathcal{A}_{\overline{D}}|} = \frac{\sum_{q \in \text{desc}(D)} kl(q, q_s)}{|\text{desc}(D)|} \tag{9}$$

which is the average value of $kl(q, q_s)$ on all deleted states $q \in \text{desc}(D)$.One can easily shows that the optimal cut set of states D (for our first problem) consists of only one state: the state q_{opt} that minimizes the average kl of its descendants. The whole set of deleted states is therefore limited to the branch $\text{desc}(q_{\text{opt}})$.

Preliminary experiments showed that this branch is often reduced to only one state in practice. Since our goal is to delete a significant number of states, we will focus on the second problem in which we want to delete at least nb states. It can be solved with a recursive algorithm. For each subtree of the PPTA (rooted in q) and for all $0 \leq i \leq nb$, the algorithm computes the optimal cut set of states $D(q, i) \subseteq \text{desc}(q)$ to delete with the constraints that $|\text{desc}(D(q, i))| > i$ and $\mathcal{L}(\mathcal{A}, \mathcal{A}_{\overline{D(q,i)}})$ is minimal. However, this algorithm has a complexity $O(nb.|\mathcal{A}|.|\Sigma|)$ which is too high for large PPTA and large values of nb (we want to be able to have nb of the order of $|\mathcal{A}|$). We present a heuristic in the next section.

5 Algorithm and Complexity

We focus here on our second problem, i.e., to find a cut set of states $D = \{q_1, .., q_k\}$ such that:

- $|\text{desc}(D)| > nb$;
- D minimizes $\mathcal{L}(\mathcal{A}, \mathcal{A}_{\overline{D}}) = \frac{\sum_{1 \leq i \leq k} |\text{desc}(q_i)|.v(q_i)}{|\text{desc}(D)|}$ with $v(q_i) = \frac{\sum_{q \in \text{desc}(q_i)} kl(q, q_s)}{|\text{desc}(q_i)|}$ which is the weighted average of the $v(q_i)$ with weights $|\text{desc}(q_i)|$.

If the q_i could be chosen without constraints in Q, this problem could be solved by:

1. For each state $q \in Q$, compute $v(q)$ and $|\text{desc}(q)|$ (these values can be computed in time $O(|\mathcal{A}|)$ using a recursive traversal of the PPTA);
2. Construct a list $L =< q_1, ... >$ by sorting the states q in increasing order of $v(q)$ in time $O(|\mathcal{A}|. \log |\mathcal{A}|)$ (notice that the first element of this list is q_{opt}, the solution of the first problem);
3. Take the shortest prefix $< q_1, ..., q_k >$ of L such that the sum of the weights of q_1,...,q_k is greater than nb.

4. Take $\{q_1, ..., q_k\}$ as D.

This algorithm is optimal and has complexity $O(|\mathcal{A}|.\log|\mathcal{A}|)$. However, in our problem, the set D must be a cut set of states, *i.e.*, there cannot exist two states p and q in D such that $p \in \text{desc}(q)$. Our heuristic is to replace the third and fourth steps by:

3'. Take the shortest prefix $< q_1, ..., q_k >$ of L such that, when considering in this prefix only the states that are not descendant of one another, the sum of their weights is greater than nb. This can be done in time $O(|\mathcal{A}|)$ by marking each descendant of each state q_i and summing only the weights of the non marked states.
4'. Take the non marked states of this prefix as D.

The complexity of this algorithm is therefore $O(|\mathcal{A}|.\log|\mathcal{A}|)$.

6 Experimentations

We will evaluate the pruning first by analyzing the increment of the KL w.r.t. the number of states pruned. Then we measure its impact on grammatical inference by comparing the result of the inference when started from a pruned PPTA and from a non-pruned PPTA.

6.1 KL behavior with respect to the number of states pruned

In order to evaluate the quality of the pruning procedure, the second algorithm is used to compute a cut set of state D for different pruning thresholds nb (minimal number of states to delete).

Fig. 2. KL w.r.t. the number of states pruned

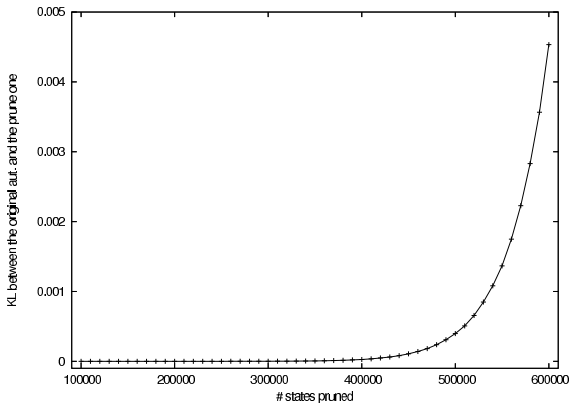


Figure 2 presents the divergence $KL(\mathcal{A}, \mathcal{A}_{\overline{D}})$ with respect to nb. The input PPTA $\mathcal{A}$ has 868,851 states (built on the wall street journal task, see section 6.3). We expect

the KL to increase monotonically. It is however noticeable that it is almost flat up to 400,000 states pruned (that is 46% of the automaton's states). The pruning is in practice efficient as it takes less than 20 seconds on a recent PC machine[1] (including Input/Output time). This value is negligible with respect to the 91 hours needed for the inference on the whole learning set (see table Sect. 6.3).

This means that the distribution $\mathcal{D}_{\mathcal{A}_{\overline{D}}}$ represented by the pruned automaton is close to the distribution represented by $\mathcal{A}$ even when a large number of the states are deleted. We thus expect to be able to use the pruned model $\mathcal{A}_{\overline{D}}$ as a replacement for the PPTA $\mathcal{A}$ built with the full training sample.

6.2 Evaluation measure

We will evaluate the pruning method in the context of language modeling. In many probabilist grammatical inference algorithms [3–6], a PPTA is built from a sample as a first step of the inference. It is then generalized by merging equivalent states. The algorithms differ in the way equivalence of states is considered.

These algorithms are evaluated through the ability of the resulting automata to parse new strings, that is by the *per symbol log-likelihood* of strings x belonging to a test sample S according to its underlying distribution:

$$LL(S) = \left(-\frac{1}{||S||} \sum_{x \in S} |S|_x \sum_{i=1}^{|x|} \log p_{\mathcal{A}}(\delta_{\mathcal{A}}(q_0, x_1 x_2 ... x_{i-1}, x_i) \right)$$

where x_i is the i-th symbol of x and $|S|_x$ its multiplicity in S. This average log-likelihood is related to the KL divergence between an unknown target distribution and the hypothesis by considering the test sample as the empirical estimates of the unknown distribution [11].

The *test sample perplexity* $PP(S)$ is most commonly used for evaluating language models. It is given by $PP(S) = 2^{LL(S)}$. The minimal perplexity $PP = 1$ is reached when the next symbol x_i is always predicted with probability 1 from the current state q_i (*i.e.* $p(\delta(q_0, x_1 ... x_{i-1}), x_i) = 1$) while $PP = |\Sigma|$ corresponds to random guessing from an alphabet of size $|\Sigma|$.

6.3 The Wall Street Journal Task

The data used are drawn from the Wall Street Journal database, a large syntactically annotated corpus subdivided in 25 sections. We followed the preprocessing used by [14]: Sections 0 to 20 were used as the training set (962,612 words), sections 21 and 22 were used as a development set (48,024 words) and sections 23 and 24 serve as the test (101,189 words). Digit numbers were replaced by a unique character. Following [14], the 10,000 most frequent words were kept and the remainder were transformed in a unique symbol `unknown`. In order to make the data more realistic for a speech to text task, the punctuation was removed and the words transformed in their lower case form. The average length of the sentences is 22 words and the size of Σ is around 10000.

We applied the pruning algorithm on the PPTA built on the whole training set (868,851 states) with different levels of pruning (number of deleted states ranging

[1] The machine used is a linux box with a 3 Ghz processor and 1 GB of memory.

from 100K to 600K). The MDI [4] algorithm is then applied on each of these pruned PPTAs. The MDI algorithm depends on a parameter α that controls the generalization rate. To choose the value of this parameter, several inferences were performed on each PPTA with α ranging from 0.00008 to 0.01. The value that gives the best perplexity on the development set is chosen. The following table gives the best value of α and the total time spent to compute all automata for all values of α.

Then, the best automaton (corresponding to the best value of α) is evaluated by its perplexity computed on the test set (see table). The size of this best automaton is also given (# states post inference).

Test set results of the inference using pruned PPTA

% states pruned	0	11%	23%	35%	46 %	57%	69%
# states pruned ($\times$1000)	0	100	200	300	400	500	600
α ($\times 10^{-4}$)	2	2	2	2	2	1	1
Total time (h)	91	72	61	47	60	20	8
$PP_{MDI}(test)$	549	550	564	547	564	567	568
# states post inference	451	463	458	422	434	1989	1863

7 Discussion

As can be seen in the above table, the test set perplexity of the inferred model does not change significantly up to a pruning level of 35% of the states. This is consistent with Fig. 2. We can also notice that the pruning has little influence over the size of the inferred automaton (at least for pruning levels below 46%).

It is worth stressing that the optimal value of the parameter α at different pruning thresholds does not change a lot. The pruning can thus be used to estimate the optimal parameter α in a less demanding situation. Only one inference will then be done on the full PPTA using the optimal setting of the pruned one.

Related work : language modeling is classically done using n-grams. As the size of the corpora increases, increasing the size of n improves the prediction. Unfortunately, the size of the model dramatically increases in n. In [15] Stolcke proposed to prune the n-gram model using an entropy based criterion similar to ours. As it is applied on n-gram, such a pruning will remove the estimation of part of sentences and not end of sentences as we do. Another main difference is the length of the removed chunk. In our case, the size of the chunk we can remove is not bounded as with n-gram. If a very long sentence appears once, our pruning method will remove the whole branch, which will not be the case with n-gram.

Regarding the prediction performances, we need to mention that the MDI algorithm is outperformed by the unpruned trigram model that obtains a test set perplexity of 165[2]. Many improvements can nevertheless be achieved at the grammatical inference level: use another algorithm [5, 6, 16], or use preprocessing techniques (*e.g.* word clustering [17], bagging [18]) or post-processing techniques (*e.g.* combining automata [5]).

[2] The trigram considered here used a Kneyser-Ney smoothing [8].

8 Conclusion and Further Work

We presented in this article an efficient way to prune a smoothed PPTA. We showed that we can dramatically reduce the size of the automaton keeping a great similarity between the distributions represented by the original automaton and the pruned one. We then showed, on a reasonable language modelling task, that the pruning can be seen as a preprocessing technique before applying grammatical inference algorithm. The performance of the learning algorithm is not changed up to a pruning level of 35% of the states of the initial automaton.

From a theoretical view point, we intend to study pruning of general automata, which leads to both computational and theoretical problems, since the efficient computation of $\mathcal{A}_{\overline{q}}$ is more complicated to achieve in the general case. From the practical point of view, it is of great interest since the pruning could be applied as a post-processing step after the inference, leading to smaller models that perform similarly.

References

1. Casacuberta, F., Vidal, E.: Machine translation with inferred stochastic finite-state transducers. Computational Linguistics **30**(2) (2004) 205–225
2. Ron, D., Singer, Y., Tishby, N.: On the learnability and usage of acyclic probabilistic finite automata. In: Proceedings of COLT 1995. (1995) 31–40
3. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Second ICGI. (1994) 139–152
4. Thollard, F., Dupont, P., de la Higuera, C.: Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In Langley, P., ed.: ICML, Morgan Kaufmann (2000)
5. Thollard, F.: Improving probabilistic grammatical inference core algorithms with post-processing techniques. In: ICML'01, Morgan Kauffman (2001) 561–568
6. Kermorvant, C., Dupont, P.: Stochastic grammatical inference with multinomial tests. In: ICGI. Volume 2484 of LNCS. (2002) 149–160
7. Vidal, E., Thollard, F., de la Higuera, C., , Casacuberta, F., Carrasco, R.C.: Probabilistic finite-state machines – Part I and II. IEEE trans. on PAMI **27**(7) (2005)
8. Goodman, J.: A bit of progress in language modeling. Technical report, Microsoft Research (2001)
9. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. on Information Systems **22**(2) (2004) 179–214
10. Clark, A., Thollard, F.: Pac-learnability of probabilistic deterministic finite state automata. JMLR **5** (2004) 473–497
11. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications. John Wiley & Sons (1991)
12. Abe, N., Warmuth, M.: On the computational complexity of approximating distributions by probabilistic automata. Machine Learning **9** (1992) 205–260
13. Carrasco, R.C.: Accurate computation of the relative entropy between stochastic regular grammars. RAIRO TIA **31**(5) (1997) 437–444
14. Charniak, E.: Immediate-head parsing for language models. In: 10th Conf. of the Association for Computational linguistic, ACL'01. (2001)
15. Stolcke, A.: Entropy-based pruning of backoff language models. In: DARPA Broadcast News Transcription and Understanding Workshop. (1998) 270–274
16. Callut, J., Dupont, P.: Learning partially observable markov models from first passage times. In: ECML. (2007) 91–103
17. Dupont, P., Chase, L.: Using symbol clustering to improve probabilistic automaton inference. In: ICGI. Volume 1433 of LNAI., Springer (1998) 232–243
18. Thollard, F., Clark, A.: Shallow parsing using probabilistic grammatical inference. In: ICGI. Volume 2484 of LNCS. (2002) 269–282